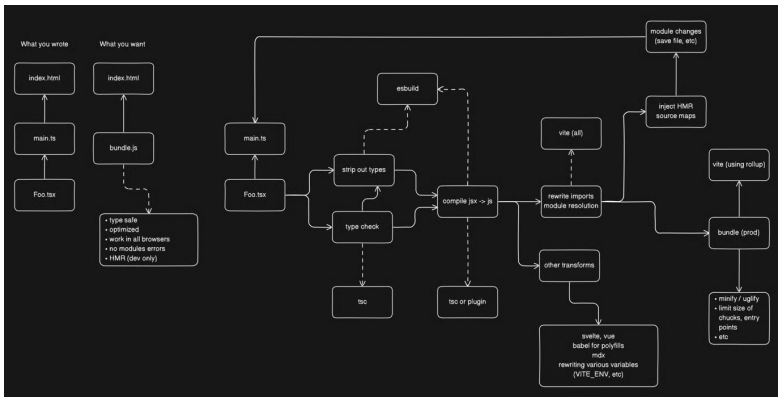
bundle

module □□

UMD (Universal Module Definition)

```
(function (global, factory) {
  // 1. CJS for nodeJS□exports □□□ nodeJS □□
  // globalThis === global
  typeof exports === 'object' && typeof module !== 'undefined' ? factory(exports) :
  // 2. CMD□SeaJS□□define □□□ seajs □□□□
   typeof define === 'function' && define.amd ? define(['exports'], factory) :
  // 3. AMD□requireJS □□□□□ exports □□
  // globalThis === self
(global = global || self, factory(global.React = {}));
}(this, (function (exports) { 'use strict';
   // factory code here...
})
```

vite



client.js

```
class HMRClient {
   // constructor and clear methods, etc
   // ...

async queueUpdate(payload) {
this.updateQueue.push(this.fetchUpdate(payload));
if (!this.pendingUpdateQueue) {
this.pendingUpdateQueue = true;
await Promise.resolve();
this.pendingUpdateQueue = false;
const loading = [...this.updateQueue];
this.updateQueue = [];
(await Promise.all(loading)).forEach((fn) => fn && fn());
}
}
async fetchUpdate(update) {
const { path, acceptedPath } = update;
const mod = this.hotModulesMap.get(path);
if (!mod) {
return;
}
let fetchedModule;
const isSelfUpdate = path === acceptedPath;
// determine the qualified callbacks before we re-import the modules
const qualifiedCallbacks = mod.callbacks.filter(({ deps }) => deps.includes(acceptedP
if (isSelfUpdate || qualifiedCallbacks.length > 0) {
const disposer = this.disposeMap.get(acceptedPath);
if (disposer)
await disposer(this.dataMap.get(acceptedPath));
try {
fetchedModule = await this.importUpdatedModule(update);
```

```
  }
  catch (e) {
  this.warnFailedUpdate(e, acceptedPath);
  }
  }
  return () => {
  for (const { deps, fn } of qualifiedCallbacks) {
  fn(deps.map((dep) => (dep === acceptedPath ? fetchedModule : undefined)));
  }
  const loggedPath = isSelfUpdate ? path : `${acceptedPath} via ${path}`;
  this.logger.debug(`[vite] hot updated: ${loggedPath}`);
  };
  }
  }

  async function handleMessage(payload) {
   switch (payload.type) {
    case 'connected':
     // ...
    case 'update':
     // ...
      await Promise.all(payload.updates.map(async (update) => {
  if (update.type === 'js-update') {


return hmrClient.queueUpdate(update);
    }
    // css-update...
   }
   // ...
  case 'custom': {
   notifyListeners(payload.event, payload.data);
   break;
  }
  // case 'full-reload' 'prune' 'error' ...
}
```

Date: 2024-03-25
Words: 350
Time to read: 1 min

Newer Older

2024-04-04
web security

2024-03-12
web motion design